Número a adivinar

Había que crear 3 clases: Jugador, arbitro y principal
El **principal** era en el que se ejecuta el hilo.
En **Jugador** Creamos el arbitro y un id, Arbitro para acceder a la jugada porque el arbitro esta encargado de manejar las jugadas. El jugador se hace una jugada aleatoria que se le pasamos al arbitro.
En **Arbitro** Creamos el número a adivinar, El turno al jugador que le toca, Árbitro dirá cuantas jugadores se juegan (lo que le ponemos). Cuando un jugador se adivina el número el árbitro se termina el juego por que es el encargado.
Hay que poner una palabra clave "synchronized" al método jugada(el que maneja toda la jugada de cada jugador), porque (en mi Ejemplo)
Tengo tres jugadores y cada jugador entra en el método para jugar para que primer jugador(El que llega a jugar primero aleatoriamente) llegue termine sus cosas y cuando sale del método entonces el segundo se mete para hacer sus cosas y sucesivamente. Así los jugadores no van a llamar al método en el mismo tiempo.
En jugada ponemos las condiciones para terminar el juego.

BANCO

Creamos 3 clases: Principal(Banco), Persona, Cuenta
Vamos a manejar que una cuenta van a controlar 3 personas.
En **Banco** creamos una cuenta, y añadir 3 personas (Por Ejemplo). Y lanzar hilos de las personas.
A la **Persona** hay que dar una cuenta y un nombre para esa cuenta. Una cuenta puede tener varias personas.
En la persona se crea una cantidad aleatoria para sacar el dinero o para ingresar el diner.
Y en método run(), cada vez que se ingresa o se reintegra, paro con un sleep(1000) para que al hilo le da tiempo a ejecutar con los nombres diferentes de la cuenta. Porque si no lo ponía me mostraba solamente la ultima persona(Ingresando y reintegrando).
En **Cuenta** hay que pasar el saldo y el saldo máximo que puede tener esta cuenta. El ingreso y reintegro se maneja en cuenta. Hay que poner "synchronized" estos dos método porque las personas van a llamar esta cuenta en el mismo tiempo. Entonces para que no tendrá colisión hay que poner esta palabra clave "synchronized". Y hay se maneja lo que se reintegra o ingresa la persona. Cuando se supera la cantidad máxima en la cuenta se para el hilo(ERROR) o Cuando se intenta sacar desde la cuenta la cantidad que no Existe también va a terminar el hilo.

Control De Stock

Hay que crear cuatro clases: Almacén, "Control de stock"(Principal), Envío, Retirada
En **Principal** creamos el almacén, Retirada y el envío, y con retirada y envío empezamos nuestros hilos.
Creamos el almacén en el que vamos a manejar la cantidad máxma y el error que puede haber Mientras Añadiendo más stock que el máxmimo o pedido más que el stock que este en almacén.
Y hay que poner synchronized a los métodos Llegadas y salidas porque el stock cuando llega o sale se modifica al stock restando o añadiendo, para que no tendrá colisión en el stock se pone la palabra clave "synchronized".
En **Envío**, le pasamos el objeto almacén para los errores, y pasamos la retirada para obtener el stock. envío maneja el stock que llega en el almacén con una cantidad entre 400 y 1000. cuando se le llega la cantidad se imprime y suma en el almacén la cantidad que ha entrado en el almacén. Y muestra la cantidad total. Y hay que hacer un sleep(800) para que se duerme para 8 horas.
En **Retirada**, se le pasamos el almacen para controlar el error. El día que va subiendo con 24 horas que esta puesto Sleep(2400). Muestra dias. Y pedido que se retira desde almacén. Y muestra el stock total. Pedido se hace una cantidad entre 2000 y 2500. y se retira desde stock del almacén
*Estos hilos se ejecutan hasta que el almacén no da error.